E-Cell Rule file Format Reference Manual

Kouichi Takahashi (shafi@sfc.keio.ac.jp)

2000/01/31

Simulation rules written in spreadsheet software are once converted to E-Cell Rule file format, and then translated to E-Cell Rule Intermediate (.eri) which can be interpreted by the E-Cell core system. In this section, we will review the structure and method to write simulation rules in E-Cell Rule format rather than in the spreadsheet software.

0.1 Benefits to writing in the E-Cell Rule file format

There are several benefits to directly writing simulation rules in E-Cell Rule format mostly owing to the fact that the files are plain texts. Among these include:

- Easy to manage especially in huge simulation models/projects.
- Full functionality of m4 preprocessor can be utilized in the rule file. The m4 preprocessor is embbed in E-Cell Rule Compiler (er2eri command).
- Other part of UNIX command suite including programming languages like sed, awk, bc, perl, sh etc. can be utilized to process the file.
- Any text editor software can be used to write/edit.
- Portable among most platforms.

0.2 Interpretation of the E-Cell Rule file

In the E-Cell system, simulation models are constructed using interconnecting objects which are subclasses of one of three primitive classes (*Primitives*), namely *Substance* class, *Reactor* class and *System* class conforming to *Structured Substance-Reactor model.* Basically a simulation rule of E-Cell system is a set of descriptions each of these corresponds to such classes of an object.

To generate, place and give parameters to simulation objects, E-Cell core system needs following information:

1. Class name

   Possible class names depend on user configuration of the simulation system.

2. Primitive type

   One of the three Primitives to which the class belongs.

3. *SystemPath*

   A path to destined System object (where to place the object).

4. *ID*

   Identifier which is unique in the System in which the object lives.

5. *name*

   A long name only for convenience of users by which behavior of the system is never affected.

6. Parameters

   Parameters are encapsulated and delivered in form of Message class objects in the simulation system. Message object is constructed by a keyword string and arbitrary number of argument strings.

The E-CELL core system reads in such information from E-CELL Rule Intermediate(.eri) file. The E-CELL Rule Intermediate file can be generated from E-CELL Rule file by rule file compiler(er2eri command).

0.3 E-CELL Rule file format

0.3.1 Syntactic and lexical conventions

Following tokens are used to specify places (paths to Systems) and simulation objects in the simulation system.

PrimitiveType One of '`substance`', '`reactor`' and '`system`'.

ClassName Any sequence of letters, digits and character "`_`".

ID Any sequence of letters, digits and any of " `!_.+-<>=` ".

Relative SystemPath Any sequence of System IDs and '`..`' separated by slash('`/`'). The sequence '`..`' points to supersystem of the system whose ID is placed in the left. (e.g. `CELL/../ENVIRONMENT` is equivalent to `ENVIRONMENT`)

'/'(RootSystem) RootSystem is a system automatically generated at every simulation and all systems must be subsystems of it.

SystemPath Relative SystemPath starting with slash(RootSystem) and containing no '..'. (e.g. `/CELL/CYTOPLASM`, `/ENVIRONMENT`)

RQEN Relatively Qualified Entry Name. A pair of Relative SystemPath and ID separated by '`:`'. (e.g. `../CYTOPLASM:C00002`)

FQEN Fully Qualified Entry Name. A pair of SystemPath and ID separated by '`:`'. (e.g. `/CELL/CYTOPLASM:C00002`)

RQPN Relatively Qualified Primitive Name. A pair of PrimitiveType and RQEN separated by ':'. (e.g. `Substance:../../CYTOPLASM:C00002`)

FQPN Fully Qualified Primitive Name. A pair of PrimitiveType and FQEN separated by ':'. (e.g. `Substance:/CELL/CYTOPLASM:C00002`)

Following other several tokens will appear in the E-Cell Rule file.

String Any sequence of letters digits, and any of "`_.+-<>=:`"

Double Quated String Any sequence of letters, digits, white space(' ') and any of "`_.+-<>=:`" quated by `'"'`.

Comment A line starting with '`#`' is a comment line.

m4 token Any tokens which can be interpreted by m4 preprocessor as m4 macros are interpreted as m4 macros.

Other tokens Single characters like '`;`','`{`','`}`' '`(`' and '`)`'.

0.3.2 Structure of the E-Cell Rule file

E-Cell Rule file consists of (1) target sentence, (2) Primitive directives, (3) other directives, (4) m4 macros (5) comments.

target sentence The target sentence specifies destined System object (current target). Every relative paths to simulation objects are calculated from the current target.

```
target [SystemPath];
```

Primitive directive Almost all part of E-Cell Rule file would be comprised of Make-Primitive directives. The directive directs the E-Cell core system to generate and send Message objects to an simulation object in the system. It can be shown as:

```
[PrimitiveType] [ClassName] ([RQEN],[name])
{
  [message-key1] [argument1] [argument2] ... [argumentn];
  [message-key2] [argument1] [argument2] ... [argumentn];
  ...
  [message-keyn] [argument1] [argument2] ... [argumentn];
}
```

where [name] is a double quated string and message-keys and arguments are strings.

In most cases the primitive directive will appear in simulation rule to describe Substance class objects each taking only a Quantity message. Following abbreviation is available for the sake of convenience:

```
substance [RQEN] "name" [quantity];
```

which is equivalent to writing:

```
substance Substance ([RQEN],"name")
{
  Quantity [quantity];
}
```

Other directives Only one type of directive, fix directive, can be used in current version of the E-Cell System. The syntax is:

```
fix [RQEN];
```

This sets a fix flag in a Substance object specified by the RQEN (i.e. quantity of the Substance will remain constant regardless of velocity given by Reactors).

m4 macro Among extensive functionality in m4 macro processor, following several features are frequently used in E-Cell Rule files.

- include([filename])
  inserts content of a file.
- define([name] , [expansion])
  defines name to expand to expansion.

m4 manual page should be referred for more detailed description of m4 macro processor.